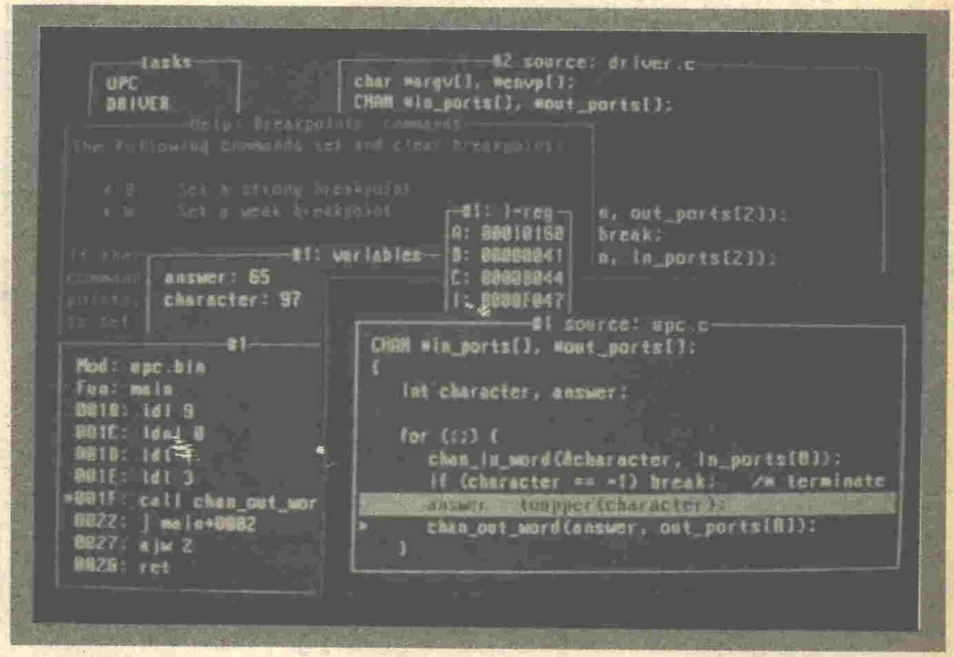
ซอฟต์แวร์ชิพ : หน้าต่าง

ดูความสามารถชิพ WM100H เคลื่อนย้ายหน้าต่างไปทั่วจอ ทั้งย่อขยายหน้าต่าง กำหนดหน้าต่างขึ้นเองได้



ในตอนที่แล้วผมได้นำเรื่องราวของซอฟต์แวร์ชิพมาเสนอ และได้เสนอชิพเบอร์ WN100H ซึ่งเป็นซอฟต์แวร์หน้าต่างพื้นฐาน แต่เนื้อหาเต็มหนักถึงการสร้างพูลดาวน์เมนู เพื่อจะนำเบอร์ WM100H มากล่าวถึงในฉบับนี้ให้สมกับที่เป็นฉบับหน้าต่างเพราะ WM100H จะเป็นเรื่องราวของหน้าต่างโดยแท้ จึงให้ชื่อตอนว่า "หน้าต่าง"

ความสามารถของ WM100H ให้ผู้ใช้สามารถเคลื่อนย้ายหน้าต่างไปได้ทั่วจอภาพ สามารถย่อหรือขยายขนาดของหน้าต่างได้ สามารถเลือกหน้าต่างที่ซ้อนทับกันอยู่ นำบานที่ต้องการขึ้นมาด้านบนหรือหมุนวนหน้าต่างนำบานบนสุดไปไว้ด้านล่าง และที่สำคัญไม่น้อยกว่ากันคือ ให้ผู้ใช้สามารถกำหนดหน้าต่างของตนเองได้ นอกเหนือไปจากที่กำหนดไว้ในโปรแกรมของผู้ประกอบ

แยกงาน

จากความสามารถของ WM100H ดังกล่าวมาจะเห็นว่าส่วนหนึ่งเป็นเรื่องราวเฉพาะของหน้าต่างหนึ่ง ๆ คือการเคลื่อนย้าย ย่อ และขยายหน้าต่าง ยิ่งผมจะทำเป็นข้อมูลแบบออปเจ็คและให้เป็นผู้สืบสกุลของ WN100H กับได้เพิ่มความสามารถอย่างอื่นอีก

ที่เหลือจะทำเป็นโปรแกรมยูทิลิตี้แต่ให้อยู่ในยูนิตเดียวกันนี้และถือว่าเป็นส่วนหนึ่งของชิพนี้ โดยตั้งชื่อกลุ่มของโปรแกรมนี้ว่า วินโดว์อะเรย์ (Window array) มี MainArrayOn เปรียบเสมือนเป็นโปรแกรมหลัก ควบคุมการทำงาน

ส่วนที่เป็นโอโอพี

ส่วนที่เป็นโอโอพี คือส่วนที่อยู่ในแบบข้อมูลออปเจ็ค WindowMove ตามรูปที่ 1 และที่เรียกว่าเป็นโอโอพีก็เป็นแต่เพียงการอ้างถึงเท่านั้น ผมเพียงแต่นำความสามารถของเทอร์โบปาสคาล 5.5 ที่ให้เรากำหนดแบบข้อมูลแบบออปเจ็คได้ มาใช้เท่านั้นครับดังที่กล่าวไว้ในฉบับก่อน

ที่เด่นก็คือเมื่อผู้เขียนโปรแกรมประกอบใช้ออปเจ็คนี้ จะเป็นการผลักภาระให้แก่ WindowMove เกือบสิ้นเชิง การติดต่อระหว่างผู้ใช้กับโปรแกรมจะติดต่อกับ WindowMove โดยตรง และไม่เกี่ยวกับโปรแกรมอื่น เว้นแต่ที่ WindowMove จะแจ้งให้ทราบดังที่ผมกำหนดฟิลด์ต่าง ๆ ไว้ดังนี้

* name เก็บชื่อหน้าต่าง
* wndsize เก็บขนาดของหน้าต่างรวมกรอบเป็น กว้าง × สูงเป็นไบต์ เพื่อจะได้ไม่ต้องคำนวณกันบ่อย ๆ
* hideflag เพื่อแสดงสถานะของ

```
UNIT  WM100H;
Interface

USES  crt,prime,WN100H;

TYPE
  UserProg    = procedure(var ch,chf : Char);


  WindowMove  = OBJECT(WindowNorm)
    name : MenuStr;  wndsize : Word;
    hideflag,firstEntry : Boolean;
    wndmenu : MenuStr;
    prgmenu : PtrToMenu;
    thiswnd : RowPtrT;
    runSubProg : UserProg;

    procedure    Init(px1,px2,py1,py2 : Byte; nme : MenuStr;
                      wmenu : MenuStr; amenu : PtrToMenu);
    procedure    ConRunSubProgTo(newprg : UserProg);
    procedure    WindowMenu(var ch : Char);
    procedure    ProgramMenu(var rstr : MenuStr);
    procedure    PutPicture(scrn : ScreenPtrT);
    procedure    GetPicture;
    procedure    ShutWindow;
                 END;
```

รูปที่ 1 แสดงแบบข้อมูลออปเจ็ค WindowMove

หน้าต่างว่าอยู่ในสถานะซ่อน (Hide) คือไม่แสดงภาพบนจอภาพ หรือแสดง (Show) ซึ่ง Init จะให้ค่าเป็น False คือไม่ซ่อน (แสดง) เมื่อเริ่มแรก

* wndmenu เป็นพอยน์เตอร์ไปยังเมนูของหน้าต่างคือ sysmenu ในรูปที่ 2 เป็นเมนูที่เตรียมไว้และมีโปรแกรมย่อยพร้อม แต่ผู้เขียนโปรแกรมประกอบอาจไม่ต้องการให้ดำเนินการเต็มที่สำหรับหน้าต่างบางบาน เช่น ไม่ต้องการให้เคลื่อนย้ายหน้าต่างคือ Move ให้ทำเมนูใหม่ไม่มีคำว่า Move และให้เมนูใหม่นี้แก่ wmenu ใน Init ผู้ใช้ก็จะเคลื่อนย้ายหน้าต่างไม่ได้

* prgmenu เป็นพอยน์เตอร์ไปยังเมนูของผู้ประกอบ ซึ่งขึ้นอยู่ว่าจะใช้หน้าต่างนี้ในงานใด และให้เป็น PtrtoMenu เพื่อให้สามารถทำเป็นเมนูหลาย ๆ ชั้นได้ ค่าเริ่มต้นที่ให้ไว้คือ progmenu ซึ่งก็เพียงแค่ให้ไว้เท่านั้น มีรายการเช่นเดียวกับ sysmenu แต่ตัด close ออก

* thiswnd เป็นพอยน์เตอร์ไปยังหน่วยความจำที่จะเก็บภาพหน้าต่างนี้ทั้งบานรวมทั้งกรอบ

```
Implementation

CONST  BlankWord = $0720;  EndPrg = #$FF;
       N  = #72;  NE = #73;  E  = #77;  SE = #81;
       S  = #80;  SW = #79;  W  = #75;  NW = #71;

       sysmenu : MenuStr =
         'WINDOW MENU,Move,Expand,Contract,Hide/sHow,eXit,clOse';


       dumMenu : MenuStr =
         'Sorry no menu for this window,  press Esc';

       prompWndName : MenuStr =
         '==== Enter window''s name ====';
```

รูปที่ 2 แสดงเมนูต่าง ๆ ที่ใช้กับหน้าต่าง

* firstEntry เป็นแฟลกสำหรับโปรแกรมประกอบที่จะตรวจตนเองว่าทำงานเป็นครั้งแรกหรือเปล่า ซึ่ง Init จะให้เป็นจริง

* runSubProg เป็นตัวแปรโปรแกรมย่อยสำหรับรันโปรแกรมที่ผู้เขียนโปรแกรมประกอบจะกำหนด ผมกำหนดไว้เผื่อใช้ในอนาคตเท่านั้น

* Init เพื่อเตรียมค่าต่าง ๆ ตามที่ผู้เขียนโปรแกรมประกอบจะกำหนด กล่าวคือ

- px1 ถึง py2 เป็นจุดโคออร์ดิเนตของหน้าต่างไม่รวมกรอบ

- nme ชื่อหน้าต่าง หากไม่มีให้เป็นสตริงว่าง

- wmenu คือเมนูหน้าต่าง หากไม่ต้องการเปลี่ยนให้ค่าเป็นสตริงว่าง

- amenu คือเมนูที่จะให้ค่าแก่ prgmenu ให้ผ่านเป็นค่าแอดเดรสมาให้ หากไม่ต้องการเปลี่ยนให้ผ่านเป็นค่า nil มา

ซึ่งใน Init จะนำค่าไปให้หรือคำนวณให้แก่ตัวแปรในออปเจ็คดังกล่าวมาแล้วเรียกใช้ Init ของ WindowNorm เพื่อเตรียมค่าต่าง ๆ ดังที่กล่าวใน WN-100H จองเนื้อที่ในหน่วยความจำเท่ากับขนาดของหน้าต่างให้แก่พอยน์เตอร์ Thiswnd และต่อ run-SubProg เข้ากับโปรแกรมเปล่า

นอกจากนี้ก่อนที่จะเรียกใช้ WindowNorm.Init จะให้ background ซึ่งเป็นตัวแปรส่วนกลางของยูนิตเก็บภาพบนจอภาพในขณะนั้น จึงเป็นภาพของจอภาพที่ยังไม่มีภาพหน้าต่างนี้ ไว้ใช้ในการเคลื่อนย้ายและเปลี่ยนขนาดหน้าต่าง

* ConRunSubProgTo เพื่อต่อ runSubProg เข้ากับโปรแกรมที่ผู้เขียน

โปรแกรมประกอบต้องการ

* ProgramMenu เป็นโปรแกรมย่อยที่จะเรียกใช้ PullDawnMenu ใน WindowNorm โดยผ่านเมนูที่เก็บในarymenu ไปให้ ก็เพียงเพื่ออำนวยความสะดวกเท่านั้นครับ

* GetPicture เก็บภาพบนจอภาพเฉพาะตรงหน้าต่างนี้รวมกรอบให้แก่ thiswnd

* PutPicture นำภาพที่เก็บไว้กับ thiswnd ไปลงบนจอภาพ ตามแต่จะกำหนดให้แก่ scrn คือเราอาจจะจัดหน่วยความจำทำหน้าที่เป็นจอภาพและดำเนินการจนเสร็จ แล้วจึงถ่ายข้อมูลให้แก่ screen ซึ่งเป็นจอภาพตัวจริงทำให้ผู้ใช้จับไม่ได้ว่ากำลังดำเนินการอะไรอยู่

* ShutWindow เมื่อต้องการปิดหน้าต่าง(ชั่วคราว) โดยเรียกใช้ WindowNorm.ShutWindow และเก็บภาพหน้าต่างด้วย GetPicture

WindowMenu

เนื่องจาก WindowMenu มีรายละเอียดที่จะกล่าวถึงมากพอสมควร จึงขอยกมาเป็นหัวข้อใหญ่ อาจจะถือได้ว่าโปรแกรมย่อยนี้เป็นโปรแกรมหลักของ Window Move เพื่อดำเนินการตามที่กล่าวมาตามความต้องการของผู้ใช้ เช่นในโปรแกรมประกอบอาจให้ผู้ใช้ได้โดยกดคีย์ F10 แล้วโปรแกรมประกอบเป็นผู้เรียกให้ (ดังจะเห็นในตัวอย่าง)

แต่ผมเห็นว่าในบางโอกาสผู้เขียนโปรแกรมประกอบ อาจต้องการใช้งานด้วยตนเองจึงได้เตรียมการไว้ดังรูปที่ 3 โดยตั้งข้อกำหนดไว้ว่าหากโปรแกรมประกอบเรียกใช้โดยให้ Ch เป็น null (ผู้ใช้ต้องการใช้) จะเรียกใช้ PullDownMenu แสดงรายการให้ผู้ใช้เลือกได้ค่าใน ch แต่ถ้าเรียกใช้โดยให้ ch มีค่าเป็นอักษรใหญ่ของรายการ (ผู้เขียนโปรแกรมเรียก) จะให้ดำเนินการโดยตรงตามค่าใน ch ดังแสดงในรูปและมีโปรแกรมย่อยสนับสนุนดังนี้ คือ

```
BEGIN  { WindowMenu }
  if  ch = null  then
    begin  PulldownMenu(5,5,@wndmenu,retstr);
           ch := retstr[1]   end;
  CASE  ch  OF
    'M' : MoveWindow;
    'E' : ExpandWindow;
    'C' : ContractWindow;
    'H' : HideWindow;
    'X' : DoExit;
    'O' : DoCloseWindow;
  END;  { case }
END;
```

รูปที่ 3 โปรแกรมหลักของ Windowmenu

* **MoveWindow** จะเคลื่อนย้ายหน้าต่างตามทิศทางของลูกศรทั้งสี่ และให้เคลื่อนย้ายในทิศทางทะแยงได้ด้วยคีย์ Home, PgUp, PgDn และ End ดังที่กำหนดเป็นค่าคงตัวไว้ในรูปที่ 2

การทำงานของโปรแกรมนี้ (สำหรับผู้ที่สนใจศึกษาโปรแกรมในซอฟต์เมล์) เริ่มด้วยการปิดหน้าต่างเพื่อเก็บภาพของหน้าต่างนี้ไว้กับ thiswnd แล้วแสดงข้อความที่บรรทัดบนสุดของจอภาพแจ้งผู้ใช้ให้ใช้คีย์ลูกศรในการเลื่อนจอภาพให้ค่า step เป็น 1 ค่านี้คือตัวคูณที่จะใช้ในการเลื่อนหน้าต่างครั้งละ 1 หรือหลายช่องอักขระ-บรรทัดแล้วแต่กรณี แล้วเข้าทำงานในวงรอบ repeat ซึ่งในวงรอบนี้จะมีการดำเนินการดังนี้

- อ่านคีย์มาตรวจ ถ้าเป็นคีย์ของเลข 1 ถึง 9 จะให้ค่าเป็นเลขจำนวนเต็มแก่ step และจะอ่านคีย์จนกว่าผู้ใช้กดคีย์ลูกศรทั้งสี่ (รวมคีย์ทะแยงอีก 4 คีย์ด้วย) ให้ค่าใน ch

- นำภาพพื้นหลังที่เก็บไว้ใน background ตั้งแต่ Init มาแสดง เท่ากับลบภาพของหน้าต่างนี้

- คำนวณหาจุดโคออร์ดิเนตของหน้าต่างใหม่ด้วยค่าทิศทางใน ch และระยะทางใน step

- นำภาพของตนตามที่เก็บไว้กับ thiswnd ออกแสดงทับ ณ ตำแหน่งใหม่ที่คำนวณได้ ด้วยวิธีการเช่นนี้ทำให้ผู้ใช้สามารถเลื่อนหน้าต่างไปในทิศทางและระยะทางที่ต้องการได้

- ให้ทำซ้ำจนกว่าผู้ใช้จะกดคีย์ Enter จึงให้เปิดใช้หน้าต่างใหม่

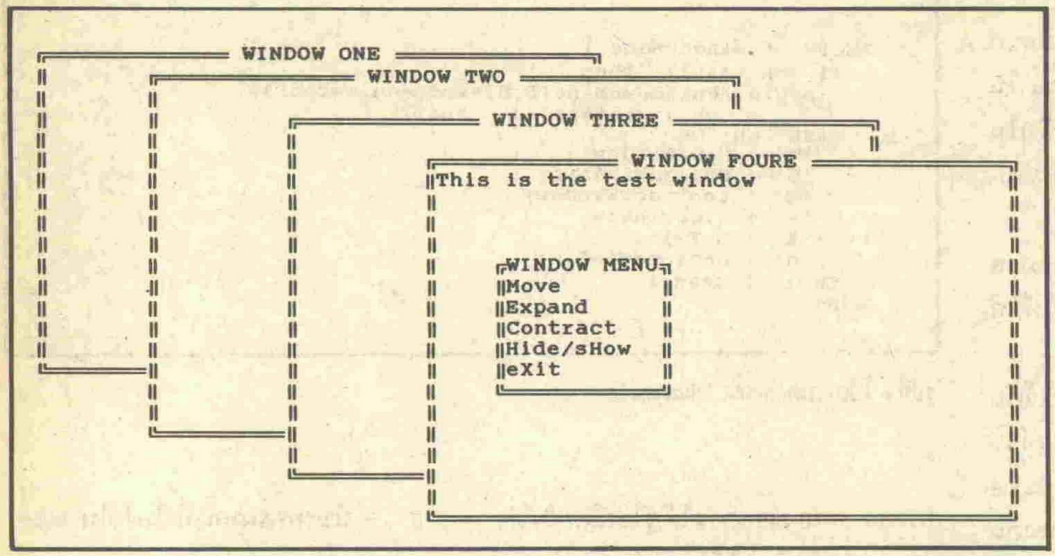
ดังผลการรันในรูปที่ 4 และ 5

* **ExpandWindow** เพื่อการขยายหน้าต่าง ดำเนินการเช่นเดียวกับ MoveWindow แตกต่างกันตรง

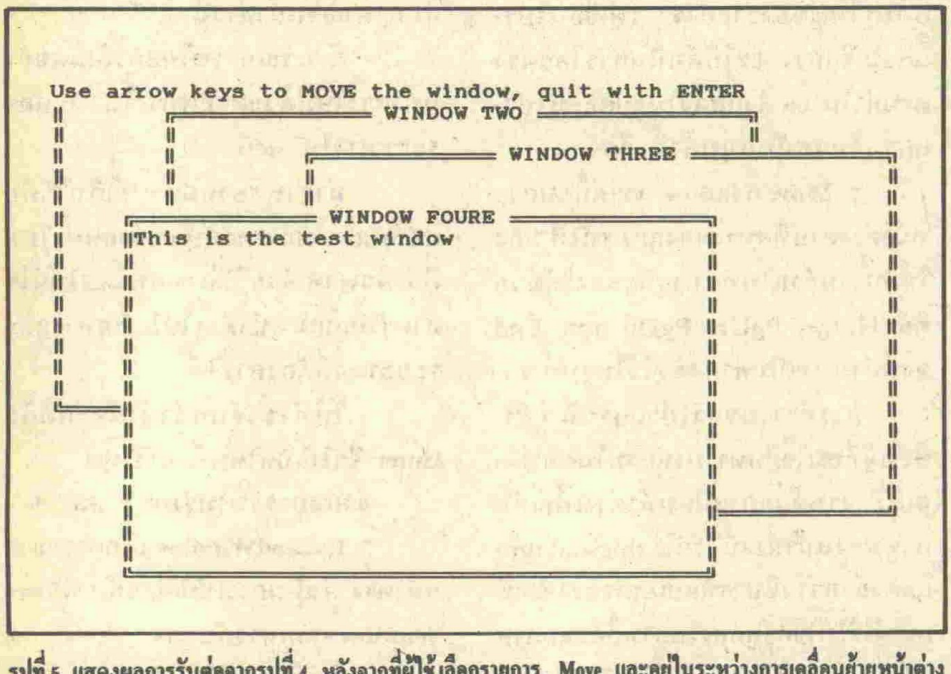
- นำค่าทิศทางใน ch กับระยะทางใน step ไปคำนวณหาค่าโคออร์ดิเนตของหน้าต่างใหม่ โดยขยายกรอบหน้าต่างด้านลูกศรออกไป

- ไม่ได้นำภาพหน้าต่างที่เก็บไว้กับ thiswnd มาลงทับ เพียงแต่ตีกรอบหน้าต่างทับเท่านั้น

- เมื่อผู้ใช้กดคีย์ Enter แล้ว จะลบภาพในกรอบหน้าต่าง การที่จะนำภาพออกแสดงใหม่ให้เป็นหน้าที่ของ



รูปที่ 4 แสดงผลการรันของโปรแกรมทดสอบ ตามรูปที่ 7 เมื่อผู้ใช้เรียกใช้เมนูหน้าต่าง



รูปที่ 5 แสดงผลการรันต่อจากรูปที่ 4 หลังจากที่ผู้ใช้เลือกรายการ Move และอยู่ในระหว่างการเคลื่อนย้ายหน้าต่าง

โปรแกรมประกอบ ซึ่งจะทราบได้โดย InfoWndWidHi ใน WindowNorm จะแจ้งความกว้างและความสูงของหน้าต่างใหม่ให้ทราบ

* **ContactWindow** เพื่อการย่อหน้าต่าง ดำเนินการเช่นเดียวกันกับ ExpandWindow แค่เป็นการลดขนาดของหน้าต่าง

* **HideWindow** ในบางครั้งผู้ใช้อาจต้องการยกหน้าต่างขึ้นบนออกชั่วครู่เพื่อดูข้อความที่หน้าต่างนั้นทับอยู่แล้วนำมาวางทับตามเดิม ผมจึงจัดรายการนี้คือ Hide/Show ไว้ให้ผู้ใช้เลือกกลับไปกลับมา (เช่น กดคีย์ F10,H แล้ว F10,H อีกครั้ง) โดยทำงานสัมพันธ์อยู่กับ hideflag ซึ่งในโปรแกรมเพียงแต่นำค่าใน background มาแสดงเมื่อผู้ใช้เรียก Hide และนำค่า thiswnd มาแสดงทับเมื่อผู้ใช้เลือก sHow เท่านั้น

* **Exit** เมื่อผู้ใช้ต้องการยุติการทำงานในหน้าต่างนี้ชั่วคราว (เช่นเพื่อไปทำงานในหน้าต่างอื่น) ในโปรแกรมจึงเพียงแต่ให้ปิดหน้าต่างนี้ด้วย Shut-Window

* **DoCloseWindow** เมื่อผู้ใช้เลือกรายการ close หมายถึงต้องการเลิกใช้หน้าต่างนี้ จึงให้ยกเลิกหน่วยความจำในชีพที่เก็บภาพของหน้าต่างนี้ตามที่ thiswnd ชี้อยู่ และลบภาพของหน้าต่างนี้บนจอภาพ เป็นหน้าที่ของผู้เขียนโปรแกรมประกอบที่จะต้องป้องกันไม่ให้ผู้ใช้นำหน้าต่างนี้มาใช้ได้อีก

โปรแกรมในวินโดว์อะเรย์

ในการใช้งานหน้าต่างหลาย ๆ บานกับจอภาพที่ใช้กับเครื่อง IBM-PC จำเป็นที่จะต้องให้หน้าต่างเหลื่อมทับกันบ้าง จึงต้องมีวิธีที่จะนำหน้าต่างที่จะเปิดใช้งานขึ้นมาอยู่ด้านบน ซึ่งผมจะดำเนินการโดยกำหนดตัวแปร และมีโปรแกรมสนับสนุนดังนี้

```
TYPE
    WndRecT   =  RECORD
       wndptr : PtrToWndMove;
       num : Byte;
             END;

    WndArrayT = ARRAY[1..20] OF WndRecT;


VAR     mainscreen : WindowMove;
        wndArray : WndArrayT;
        dumwnd : PtrToWndMove;
        namestr : MenuStr;
        wIdx : Byte;
```

รูปที่ 6 แสดงการกำหนดแบบข้อมูลและตัวแปรของวินโดว์อะเรย์

ตัวแปร

ตามรูปที่ 6 ได้กำหนดแบบข้อมูลเรคอร์ด WndRect มีฟิลด์ wndptr เป็นข้อมูลแบบ PtrToWndMove คือเป็นพอยน์เตอร์ไปยังข้อมูลแบบออปเจ็ค WindowMove ที่พึ่งกล่าวถึงกับฟิลด์ num เพื่อเก็บหมายเลขของหน้าต่าง แล้วนำไปกำหนดแบบข้อมูล WndArrayT เป็นอะเรย์ขนาด 20 ช่องของ WndRect นั่นคือผู้ใช้ยูนิตนี้จะกำหนดหน้าต่างได้ไม่เกิน 20 หน้าต่าง ซึ่งผมเห็นว่ามากพอสมควรแล้ว จากนี้ได้นำไปกำหนดตัวแปรดังนี้คือ

- mainscreen เป็นตัวแปรแบบออปเจ็ค WindowMove เพื่อแสดงเมนูในการซ้อนทับหน้าต่างให้เลือก

- wndArray เป็นตัวแปรอะเรย์แบบ WndArrayT ซึ่งจะใช้เนื้อที่ในหน่วยความจำเพียง 20×5 = 100 ไบต์เท่านั้น เพราะได้กำหนดฟิลด์ Wndptr ไว้เป็นพอยน์เตอร์

- dumwnd เป็นพอยน์เตอร์ไปยังข้อมูลแบบ WindowMove เพื่อการกำหนดหน่วยความจำสำหรับข้อมูลแบบนี้ในฮีพ

- wIdx เพื่อเป็นอินเด็กซ์ของอะเรย์ WndArray ซึ่งจะให้ค่าเริ่มต้นเป็นศูนย์

- namestr เพื่อเก็บชื่อหน้าต่างในอะเรย์ทั้งหมด

นอกจากนี้ยังได้กำหนดตัวแปรสำหรับใช้งานในวินโดว์อะเรย์ ไว้ในส่วน Interface ของยูนิตนี้ เพื่อให้โปรแกรมที่ใช้งานยูนิตนี้ตรวจสอบได้คือ

- actwnd สำหรับเก็บพอยน์เตอร์ไปยังหน้าต่าง (คือตัวแปรออปเจ็กแบบ WindowMove) ซึ่งเปิดใช้อยู่ในขณะนั้น

- numOfOrgWnd เก็บจำนวนหน้าต่างที่ผู้เขียนโปรแกรมประกอบกำหนด คือที่ยกเลิกไม่ได้

โปรแกรมย่อย InitWndToArray

เมื่อผู้เขียนโปรแกรมประกอบต้องการกำหนดให้หน้าต่างบานใดอยู่ในอะเรย์จะต้องกำหนดตัวแปรหน้าต่างคือตัวแปรแบบออปเจ็ค WindowMove ขึ้นก่อนแล้วเรียกใช้โปรแกรมย่อยนี้ โดยผ่านค่าแอดเดรสของตัวแปรหน้าต่างไปให้ พร้อมด้วยโคออร์ดิเนตของจุดมุมและชื่อหน้าต่าง ดังแสดงในรูปที่ 7 ซึ่งเป็นโปรแกรมสำหรับทดสอบการใช้งานยูนิตนี้

ซึ่งในโปรแกรมย่อย InitWndToArray จะดำเนินการดังนี้ คือ

- ปิดหน้าต่างในอะเรย์ ก่อนหน้านี้หากมี แล้วเพิ่มค่า wIdx ขั้นหนึ่ง

- เตรียมหน้าต่างใหม่ด้วย Init ของ WindowMove ซึ่งจะเป็นการเปิดใช้หน้าต่างนี้ไปในตัว

- ให้ wndptr ของช่องที่ wIdx คือช่องใหม่ชี้ไปที่ตัวแปรหน้าต่างใหม่นี้และให้หมายเลขหน้าต่างมีค่าเท่ากับ wIdx

```
var   window1,window2,window3,window4,window5 : WindowMove;
      ch : Char;
      retstr,prompl : MenuStr;
      note : NoteArray;  idx : Word;

begin  { MAIN }

  InitWndToArray(@window1,2,40,2,15,'WINDOW ONE');

  InitWndToArray(@window2,10,50,3,18,'WINDOW TWO');

  InitWndToArray(@window3,20,60,5,20,'WINDOW THREE');

  InitWndToArray(@window4,30,70,7,22,'WINDOW FOURE');

   numOfOrgWnd := 4;  idx := 1;
   repeat
     MicroEdit(note,idx,1000,ch);
     MainArrayOn(ch);
   until  ch = 'Q';

Write('End of program press ENTER');  ReadLn;
end.
```

รูปที่ 7 โปรแกรมหลักของโปรแกรมที่ใช้ทดสอบยูนิต WM100H

- ให้ actwnd ชี้ไปที่หน้าต่างใหม่นี้เช่นกัน ดังนั้นการอ้างถึงหน้าต่างนี้จึงทำได้ 3 วิธี สมมติว่าเป็น window1 (รูปที่ 7) จะอ้างถึงได้ด้วย window1 (คือตัวของมันเอง) และ wndArray [widx].wndptr กับ actwnd

- ท้ายสุดเป็นการเพิ่มชื่อของหน้าต่างใหม่นี้กับ namestr เพื่อการเลือกหน้าต่างด้วยชื่อ

โปรแกรมย่อย MainArrayOn

เมื่อผู้เขียนโปรแกรมประกอบต้องการให้ผู้ใช้เลือกหน้าต่างในอะเรย์ (เช่นให้กดคีย์ F10) จะต้องเรียกใช้โปรแกรมย่อยนี้ โดยผ่านค่า ch มาให้ด้วยและจะเป็นการทำงานใน mainscreen ซึ่งเป็นหน้าต่างเต็มจอภาพ(ไม่มีกรอบหน้าต่าง) ตามที่ส่วน begin end. ของยูนิตนี้เตรียมไว้ให้ดังรูปที่ 8

และตามรูปนี้จะเห็นว่าได้ให้ wndmenu และ prgmenu ของ mainscreen มีค่าเป็น dumMenu และ wndArrayMenu ตามลำดับ

ซึ่งเมื่อเรียกใช้งาน MainArrayOn ตามรูปที่ 9 จะมีการดำเนินการดังนี้คือ

- ปิดหน้าต่างที่เปิดใช้อยู่ในขณะนั้น

- ในวงรอบ repeat จะตรวจว่า ch เป็นอักขระ null หรือ 'C' หรือ F10 หรือไม่ ถ้าใช่ก็จะเปิดหน้าต่าง mainscreen แล้วเรียกใช้ ProgramMenu เป็นการเรียกใช้ PullDownMenu แสดงรายการให้ผู้ใช้เลือก ได้ค่าคืนมาใน rstr ซึ่งนำมาถ่ายค่าให้แก่ ch

- นำมาตรวจด้วย CASE แยกการทำงานตามที่ผู้ใช้ต้องการดังจะกล่าวต่อไป

- จะทำงานอยู่ในวงรอบนี้จนกว่าค่าใน ch ไม่เท่ากับ C นั่นคือหากเลือกรายการ Cycling จะคงทำงานอยู่ในวงรอบนี้ ค่าคีย์นอกนั้นจะออกจากวงรอบทันที รวมทั้งเมื่อผู้ใช้กดคีย์ Esc แสดงว่าไม่ต้องการเลือกรายการใดด้วย

โปรแกรมสนับสนุน

WindowMove เพื่อดำเนินการตามที่กล่าวมาตามความต้องการของผู้ใช้ เช่นในโปรแกรมประกอบอาจให้ผู้ใช้เรียกใช้ได้โดยกดคีย์ F10 แล้วโปรแกรมประกอบเป็นผู้เรียกให้ (ดังจะเห็นในตัวอย่าง)

เพื่อให้การดำเนินการเป็นไปตามที่กล่าวมา ได้จัดโปรแกรมย่อยสนับสนุนดังนี้คือ

* **DispAllWindow** นำภาพของหน้าต่างทุกบานออกแสดงทางจอภาพ โดยให้หน้าต่างแรก (wIdx = 1) อยู่ชั้นล่างสุด และหน้าต่างสุดท้ายอยู่ชั้นบนสุด ในการนี้ได้ให้เก็บภาพของหน้าต่างแรกจนถึงรองสุดท้ายไว้กับ backgraond ก่อน แล้วจึงถ่ายให้แก่จอภาพหลังจากนี้จึงนำภาพหน้าต่างสุดท้ายคือที่จะอยู่ชั้นบนสุดมาลง

* **Cyclewindow** เมื่อผู้ใช้เลือกรายการ Cycling เพื่อหมุนวนหน้าต่างบนสุดไปไว้ล่างสุด และนำหน้าต่างชั้นรองลงไปมาเป็นชั้นบนสุด จะให้มาทำงานที่โปรแกรมนี้โดยเก็บข้อมูลในอะเรย์ wndArray ชั้นบนสุดไว้กับ tem เลื่อนข้อมูลในอะเรย์ทั้งหมดขึ้นบน 1 ช่อง แล้วให้ช่องที่ 1 มีค่าเป็น tem ดังนั้นเมื่อให้ DispAllWindow ใหม่ก็จะเป็นการหมุนวนหน้าต่าง

```
BEGIN
  cls(@screen);  numOfOrgWnd := 255;
  wIdx := 0;  namestr := 'WINDOW NAME';
  mainscreen.Init(1,80,1,25,' ',dumMenu,@wndArrayMenu);
END.
```

รูปที่ 8 แสดงส่วน begin end. ของยูนิต WM100H

```
PROCEDURE  MainArrayOn(var ch : Char);
VAR   rstr : MenuStr;
BEGIN
  actwnd^.ShutWindow;
  REPEAT
    IF  (ch = null) or (ch = 'C') or (ch = F10)  THEN
      BEGIN
      with  mainscreen  do
        begin  Window(wx1,wy1,wx2,wy2);  GotoXY(curx,cury);
               ProgramMenu(rstr);
               if  rstr = nullstr  then  ch := Esc
               else  ch := rstr[1]  end;
      END;

    case  ch  of
     'C' : CycleWindow;
     'S' : SelectWindow;
     'N' : NewWindow;
     'R' : RenameTopWindow;
     'X' : CycleWindow;              { from eXit  in window menu }
     'O' : CloseTopWindow;
    end  { case }
  UNTIL  (ch <> 'C');
  actwnd^.ReOpenWindow;
END;
```

รูปที่ 9 โปรแกรม MainArrayOn

* **SelectWindow** ในบางโอกาสผู้ใช้ อาจต้องการเลือกหน้าต่างบานใดบานหนึ่งขึ้นมาอยู่ชั้นบนสุดโดยตรง ให้เลือกรายการ Select เมื่อเข้าทำงานในโปรแกรมนี้ จะแสดงรายชื่อหน้าต่างที่เก็บไว้ใน namestr ให้ผู้ใช้เลือกด้วย GetOptByNum ได้หมายเลขของหน้าต่างกลับมาใน num

การดำเนินการต่อไปเป็นการเรียงลำดับข้อมูลในอะเรย์ wndArray ให้หน้าต่างหมายเลข num ขึ้นมาอยู่ชั้นบนสุดด้วยวิธีดังที่เคยกล่าวในไมโครฯ 42 เสร็จแล้วให้แสดงภาพหน้าต่างทั้งหมดใหม่ด้วย DispAllWindow

* **NewWindow** ผู้เขียนโปรแกรมประกอบอาจให้ผู้ใช้กำหนดหน้าต่างของผู้ใช้ได้ เช่น ให้ใช้เป็นบันทึกซึ่งต้องใช้รวมกับโปรแกรมประเภทเวิร์ดโปรเซสเซอร์หรือเอดิเตอร์ จึงได้เตรียมรายการนี้ไว้ให้และใช้โปรแกรมนี้ดำเนินการ ซึ่งจะให้ผู้ใช้กำหนดชื่อหน้าต่างใหม่หรือใช้ที่กำหนดไว้ให้คือ "NOTE PAD" แล้วกำหนดหน้าต่างให้

ขอให้สังเกตุด้วยว่าในการกำหนดหน้าต่างให้ผู้ใช้นี้ ได้กำหนดข้อมูลหน้าต่าง (คือข้อมูลแบบ WindowMove) ไว้ในฮีพ แต่สำหรับหน้าต่างของผู้เขียนโปรแกรมประกอบ (คือ Window1 ถึง Window4 ตามรูปที่ 7) ให้กำหนดตัวแปรก่อน ซึ่งหน้าต่างของผู้เขียนโปรแกรมประกอบอาจกำหนดไว้ในฮีพได้เช่นกัน

* **RenameTopWindow** หากผู้ใช้ต้องการเปลี่ยนชื่อหน้าต่างของผู้ใช้เองก็ให้ทำได้โดยเลือกรายการ Rename ซึ่งจะมาทำงานในโปรแกรมนี้ เริ่มด้วยตรวจสอบว่าเป็นหน้าต่างของผู้ใช้หรือไม่ ถ้าใช่จึงจะให้เปลี่ยนชื่อหน้าต่างให้ โดยแสดงชื่อหน้าต่างเดิมให้ผู้ใช้ป้อนใหม่ด้วย Prompter อ่านชื่อมาแก้ทั้งในข้อมูลหน้าต่าง และใน namestr แล้วตีกรอบหน้าต่างด้วยชื่อใหม่และเก็บภาพของหน้าต่างนี้ไว้

- **CloseTopWindow** สำหรับหน้าต่างที่ผู้ใช้กำหนดขึ้นใหม่ สามารถจะยกเลิกได้โดยเลือกรายการ close ซึ่งจะมาทำงานในโปรแกรมนี้ ตรวจสอบว่าเป็นหน้าต่างของผู้ใช้หรือไม่ ถ้าใช่จึงจะดำเนินการต่อไปคือ ยกเลิกหน่วยความจำที่เป็นข้อมูลของหน้าต่างนี้ในฮีพทั้งหมด ถอนชื่อหน้าต่างนี้ใน namestr กับลดค่า wIdx ลงหนึ่ง เท่ากับตัดหน้าต่างนี้จาก namestr และอะเรย์ wndArray

ทำให้ผู้ใช้ไม่สามารถนำกลับมาใช้ได้อีก เสร็จแล้วลดหมายเลขหน้าต่างของผู้ใช้ที่มีหมายเลขมากกว่าที่ถูกยกเลิกลงหนึ่ง เมื่อให้แสดงภาพหน้าต่างทั้งหมดใหม่ ก็จะเป็นการถอนหน้าต่างนี้จากจอภาพ

โปรแกรมทดสอบ

ผมเขียนโปรแกรมทดสอบไว้ในไฟล์ TWM100.PAS ซึ่งมีโปรแกรมหลักดังแสดงในรูปที่ 7 และมีโปรแกรมสนับสนุนคือ

MicroEdit

เป็นโปรแกรมง่าย ๆ ที่จะอ่านคีย์มาตรวจสอบ ถ้าไม่ใช่พวกคีย์คอนโทรลกับคีย์หน้าที่พิเศษก็จะนำภาพอักขระนั้นออกแสดงบนจอภาพ และนำไปเก็บไว้ในตัวแปรอะเรย์ note ที่กำหนดไว้ให้เก็บอักขระได้เพียง 1000 อักขระ ความจริงแล้วโปรแกรมนี้ทำหน้าที่เทียบเท่ากับ Line editor เท่านั้น ที่เหนือกว่าคือสามารถกดคีย์ back space ไปลบข้อความที่บรรทัดบนได้ นอกจากนี้คือ

- หากผู้ใช้กดคีย์ F9 จะเรียกใช้ WindowMenu ของยูนิต TM100H ทำให้ผู้ใช้สามารถเลื่อนและปรับแต่งขนาดของหน้าต่างได้โดยอัตโนมัติ

- หากผู้ใช้กดคีย์ F8 ผมให้เรียกใช้โปรแกรม Disk Operation ซึ่งในโปรแกรมนี้จะเรียกใช้ ProgMenu เพื่อแสดงเมนูให้ผู้ใช้เลือก ได้คีย์ที่กดในrstr แต่ผมไม่ได้นำไปดำเนินการใด

ตรงนี้ครับถ้าเป็นงานจริงและสมมติว่า เป็นงานที่จะดำเนินการกับดิสค์จริงต้องเขียนเมนูใหม่ให้มีคำที่จะดำเนินการกับไฟล์ในดิสค์เช่น Load, Save, Copy, Rename เป็นต้น และให้แอดเดรสของเมนูนี้แก่ prgmenu ก่อนและเตรียมโปรแกรมสนับสนุนการทำงานดังกล่าวมา

- หากผู้ใช้กดคีย์ F10 จะออกจากการทำงานใน MicroEdit และให้ ch มีค่าเป็น F10 ซึ่งในโปรแกรมหลักจะเรียกใช้ MainArrayOn เพื่อแสดงเมนู WndArrayMenu ให้ผู้ใช้เลือกหน้าต่าง, ยกเลิกหน้าต่าง และอื่น ๆ ดังที่ได้กล่าวมา

เรื่องน่าขำคือในโปรแกรมหลัก (รูปที่ 7) ผมใช้ตัวแปร note ตัวเดียวกับทุกหน้าต่าง ดังนั้น เมื่อคุณป้อนข้อความในแต่ละหน้าต่าง ข้อความบนจอภาพจะไม่ต่อกัน คือแยกกันอยู่ในแต่ละหน้าต่าง แต่ข้อมูลใน note จะต่อกัน ทดลองดูได้ครับ โดยย่อหรือขยายหน้าต่างบานใดบานหนึ่ง ซึ่งใน MicroEdit จะแสดงข้อความทั้งหมดใหม่ด้วย FillWindow

FillWindow

จากที่กล่าวแล้วในหัวข้อ ExpandWindow ว่าเมื่อมีการลดหรือขยายหน้าต่างเป็นหน้าที่ของโปรแกรมประกอบ

จะต้องแสดงข้อความในหน้าต่างใหม่ โดยติดต่อกับ InfoWndWidHi แต่ในโปรแกรมนี้ผมดำเนินการโดยตรงด้วยการตรวจคีย์ที่ผู้ใช้กดใน MicroEdit และเรียกใช้โปรแกรมนี้

ซึ่งผมได้เขียนโปรแกรม FillWindow ไว้อย่างง่าย ๆ คือจะแสดงข้อความเดิมจากบรรทัดแรกไปจนเต็มหน้าต่างเท่านั้น (เพราะต้องการให้ปรับปรุงดังจะกล่าวต่อไป) ดังนั้น ถ้าขนาดของหน้าต่างสั้นกว่าของเดิมจะแสดงข้อความได้ไม่หมด

Readch

ตามที่กล่าวว่า MicroEdit นำภาพอักขระออกแสดงบนจอภาพนั้น ผู้ที่ดำเนินการจริงคือ Readch ซึ่งนอกจากจะแสดงภาพแล้ว ยังลบภาพเมื่อผู้ใช้กดคีย์ back space และดำเนินการเกี่ยวกับการขึ้นบรรทัดใหม่ด้วย

ยังไม่ต่อกัน

คุณผู้อ่านบางท่านอาจสงสัยว่าผมสร้างซอฟต์แวร์ชิพมาสองเบอร์แล้ว ยังไม่เห็นนำมาต่อกันดังเช่นที่เกริ่นไว้ในตอนที่แล้ว ยังครับตอนนี้ยังเป็นเพียงซอฟต์แวร์ในกลุ่มเดียวกัน ที่จะนำมาต่อกันได้จะอยู่ในฉบับต่อไปครับ

ที่ว่าสนุก

ผมคิดว่าจะให้เรื่องนี้ได้สนุกกับหน้าต่างสมกับที่เป็นฉบับหน้าต่าง แต่ถ้าคุณเพียงแค่อ่านบทความของผมคงไม่สนุกแน่ การทดลองรันโปรแกรมในซอฟต์เมล์ ก็ไม่ทราบเหมือนกันว่าจะสนุกหรือไม่ ที่ผมคิดว่าน่าจะสนุกคือ

* **ศึกษาโปรแกรมของผม** (ซึ่งผมไม่ได้คิดว่าเขียนดีเด่นแต่อย่างไร แต่ก็น่าศึกษา) แล้วปรับปรุงให้เป็นไปตามที่คุณต้องการ

* **ในโปรแกรมทดสอบ** ขอให้คุณทำเป็นโครงงานว่าจะกำหนดหน้าต่างขึ้น 2 หน้าต่าง

- หน้าต่างหนึ่งจะอ่านข้อมูลจากเท็กซ์ไฟล์ ออกแสดงและให้ผู้ใช้กดคีย์ PgUp หรือ PgDn เพื่อดูข้อมูลหน้าก่อนหรือหน้าหลังเช่นเดียวกับโปรแกรมพวก README.COM แต่ให้ผู้ใช้ป้อนชื่อไฟล์ที่ต้องการอ่านได้ด้วย

- อีกหน้าต่างหนึ่ง เพื่อให้ป้อนข้อมูลและเก็บลงไฟล์ได้ เช่นเพื่อการคัดลอกข้อความจากหน้าต่างแรก

การดำเนินการตามโครงการดังกล่าวมานี้ ต้องปรับปรุงโปรแกรมหลายแห่งครับ คือ

- ต้องเพิ่มเมนูและโปรแกรมสำหรับการ load และ save ดังกล่าวในหัวข้อ Micro Edit

- เพิ่มตัวแปรสำหรับนับบรรทัดทุกครั้งที่ผู้ใช้กดคีย์ Enter

- ในการเก็บข้อมูลลง note เมื่อผู้ใช้กดคีย์ Enter นั้นผมเก็บ แต่ carriaqe return ต้องเพิ่ม line feed ด้วย และปรับปรุง FillWindow ให้สอดคล้องกัน ทั้งนี้เพื่อให้เหมือนกับไฟล์ในเทอร์โบปาสคาล จะได้ใช้ด้วยกันได้

- ต้องเพิ่มตัวแปร note สำหรับแต่ละหน้าต่างและควรเป็นแบบพอยน์เตอร์ไปยังอะเรย์ของอักขระ

โครงงานนี้นอกจากเพื่อการศึกษาแล้ว คิดว่าน่าจะใช้ประโยชน์ได้เป็นอย่างดีด้วย ผมคิดจะทำเฉลยให้แต่คงต้องอยู่ในซอฟต์เมล์แผ่นต่อไป เพราะผมยังไม่ได้เขียนโปรแกรมเหมือนกัน และเมื่อปิดต้นฉบับนี้ ผมก็ต้องคลุมเครื่องไปเดือนเศษเพื่อไปทำงานอื่นก่อน รวมทั้งบทความตอนหน้าคงจะต้องเว้นไว้ 1 ฉบับด้วยครับ

สนใจอิเล็กทรอนิกส์
ติดตาม
เมมิคอนทิเทอร์
อิเลคทรอนิคส์
ได้ทุกเดือน

- พบกับความแปลกใหม่ของเทคโนโลยี
- ลงมือสร้างโครงงานเอาไว้ใช้ประโยชน์ด้วยตัวคุณเอง
- อ่านเรื่องน่าสนใจให้คุณได้ลึกซึ้งและกว้างขวางในแวดวงวิชาการ
- เสริมความรู้ทางด้านฮาร์ดแวร์และซอฟต์แวร์ที่เกี่ยวข้อง
- ให้รายละเอียดทางวิชาการอย่างถูกต้องและชัดเจน
- รับรู้ข่าวคราวในแวดวงวิทยุสมัครเล่น, เครื่องเสียง
- คอลัมน์ประจำฉบับน่าสนใจและติดตาม
- สินค้า แปลกๆ ใหม่ๆ จากร้านค้าที่รวบรวมไว้ให้มากกว่าใคร

เปิดดูกันวันนี้
แล้วคุณจะรู้ถึงความหมาย
บนแผงหนังสือทั่วไป